DOCUMENTAZIONE ESAME 9 maggio 2022

TRACCIA

Esame di Computer System Design /Calcolatori Elettronici 2 – prof. Mazzocca

Prova del 9 maggio 2022

Un sistema è composto da 3 unità, A, B e C, tra loro collegate mediante due periferiche parallele che interconnettono A con B e A con C rispettivamente. Il sistema opera in due fasi successive come descritto di seguito:

- Fase 1) Inizialmente, A riceve da B un messaggio di N caratteri (con N non definito inizialmente), il cui primo carattere è uguale proprio a N (la lunghezza del messaggio). La ricezione da B deve essere gestita con il meccanismo delle interruzioni. In questa fase, si assume che C non possa interrompere A.
- Fase 2) Successivamente (dopo aver completato la ricezione del primo messaggio da B), A riceve altri T (con T noto) messaggi di lunghezza N, ricevendo un carattere da B e uno da C in modo alternato (iniziando da B).

Per la fase 2 è possibile considerare 2 diverse ipotesi di funzionamento (lo studente scelga quella di riferimento):

1) **Ipotesi semplificativa**: B e C si coordinano esternamente, e B ha una priorità superiore rispetto a C. Vengono prima inviati tutti i messaggi da B e poi tutti i messaggi da C.
2) **Ipotesi standard**: non è possibile stabilire a priori l'ordine di arrivo dei caratteri da B e C; A deve gestire in SW l'alternanza.

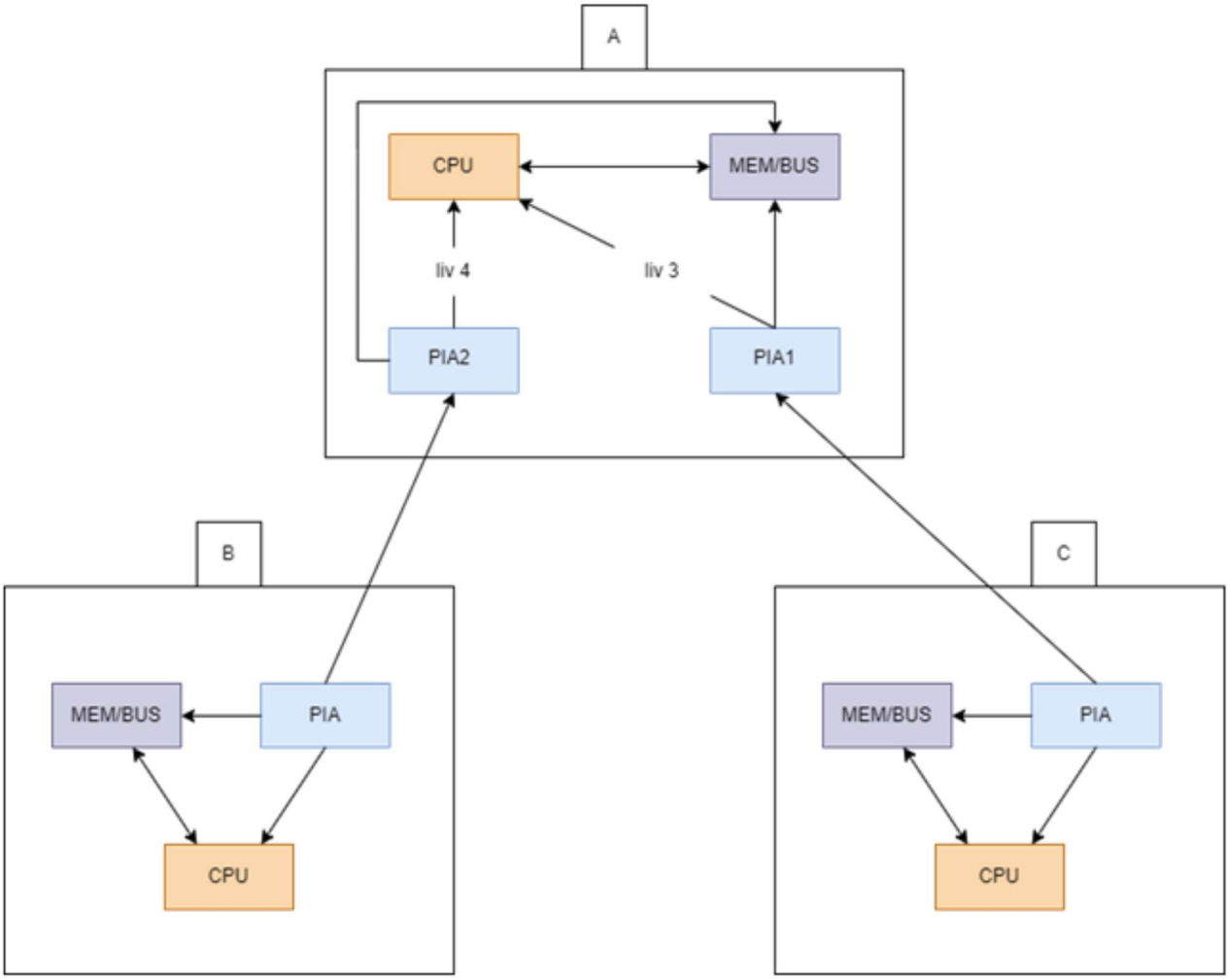
Si progetti e implementi **l'unità A** specificando:

1) *Architettura complessiva*: rappresentazione grafica schematica dell'architettura complessiva del sistema, in termini dei componenti di ciascuna unità (CPU, memoria, bus, dispositivi) e delle relative interconnessioni, in cui siano evidenziati i principali collegamenti e le linee di interruzione previste.
2) *Protocolli*: diagrammi temporali che rappresentino i principali protocolli di comunicazione utilizzati fra i dispositivi (ad es. i protocolli utilizzati per la scrittura e/o la lettura su/da periferica parallela).
3) *Mappa della memoria*: rappresentazione grafica schematica del contenuto della memoria RAM e ROM con riferimento alle aree dati e codice del programma implementato e al vettore delle eccezioni (solo per la specifica unità richiesta).
4) *Descrizione di alto livello del programma implementato*: descrizione, mediante diagramma a blocchi o pseudocodice, dei principali passi effettuati in ciascuno dei moduli software che compongono il programma (si richiede cioè un diagramma separato per il "main" e per ciascuna ISR prevista). In tale descrizione, lo studente deve specificare chiaramente le assunzioni fatte circa il comportamento delle periferiche, ad esempio legato alla gestione di possibili "conflitti" sull'accesso a dati globali e/o alla gestione di possibili "sovrapposizioni" di messaggi dovute alla diversa velocità di elaborazione dei dispositivi coinvolti.
5) *Implementazione*: codice Assembly Motorola 68000 per il sistema progettato. Gli studenti sono invitati a inserire commenti nel codice almeno nelle parti salienti (ad esempio, nella configurazione delle periferiche e nell'utilizzo di variabili globali) per favorire una migliore leggibilità e comprensione dell'elaborato.

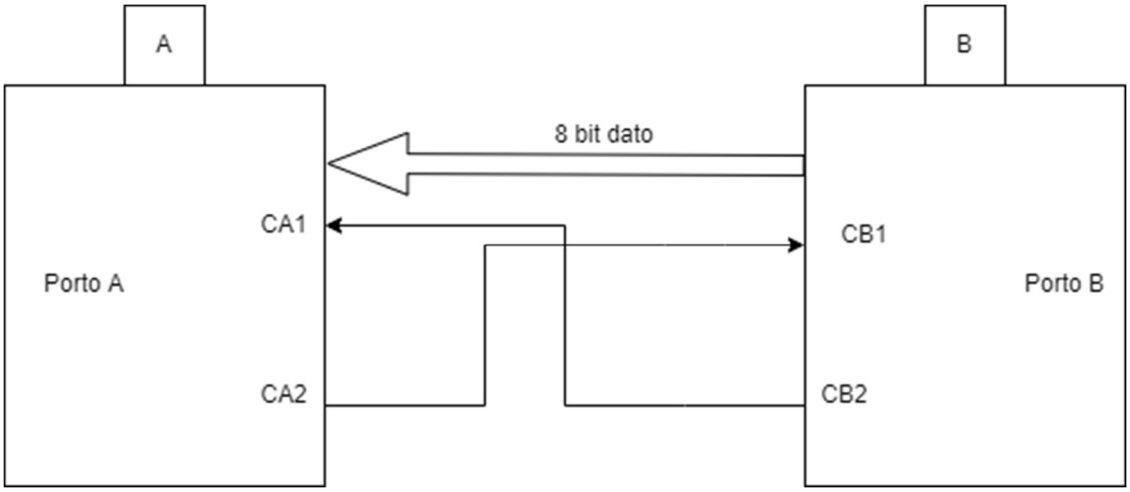
Dopo aver sviluppato l'intero progetto, si illustri come cambierebbero l'architettura complessiva e la logica del driver se venisse inserito un componente DMA per la comunicazione fra A e i nodi B e C.

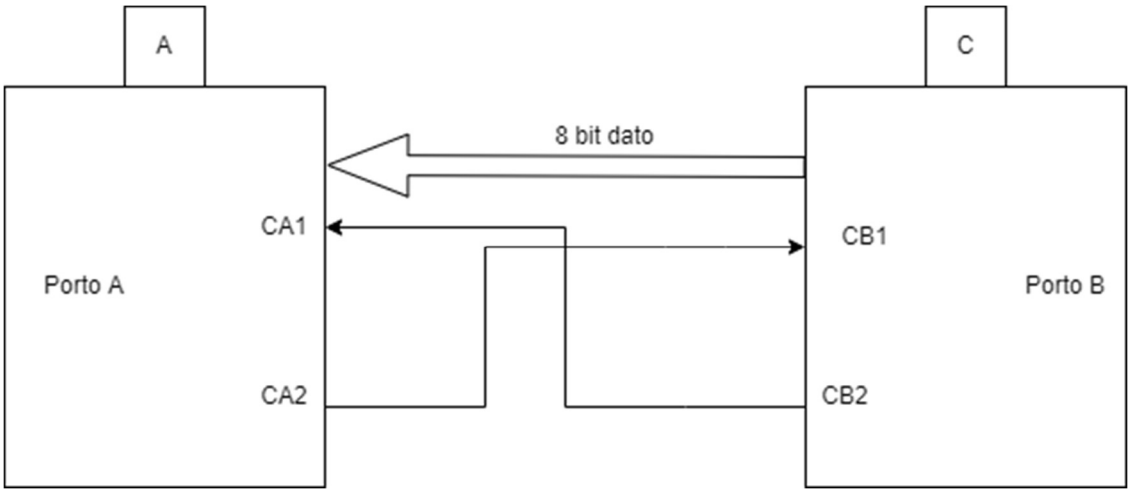
Nota: non è richiesta l'implementazione completa di un nuovo programma, ma lo studente dovrà indicare schematicamente le principali modifiche necessarie al codice assembly già prodotto (è preferibile a tale scopo indicare a parte gli stralci di codice da inserire ove necessario).

ARCHITETTURA COMPLESSIVA

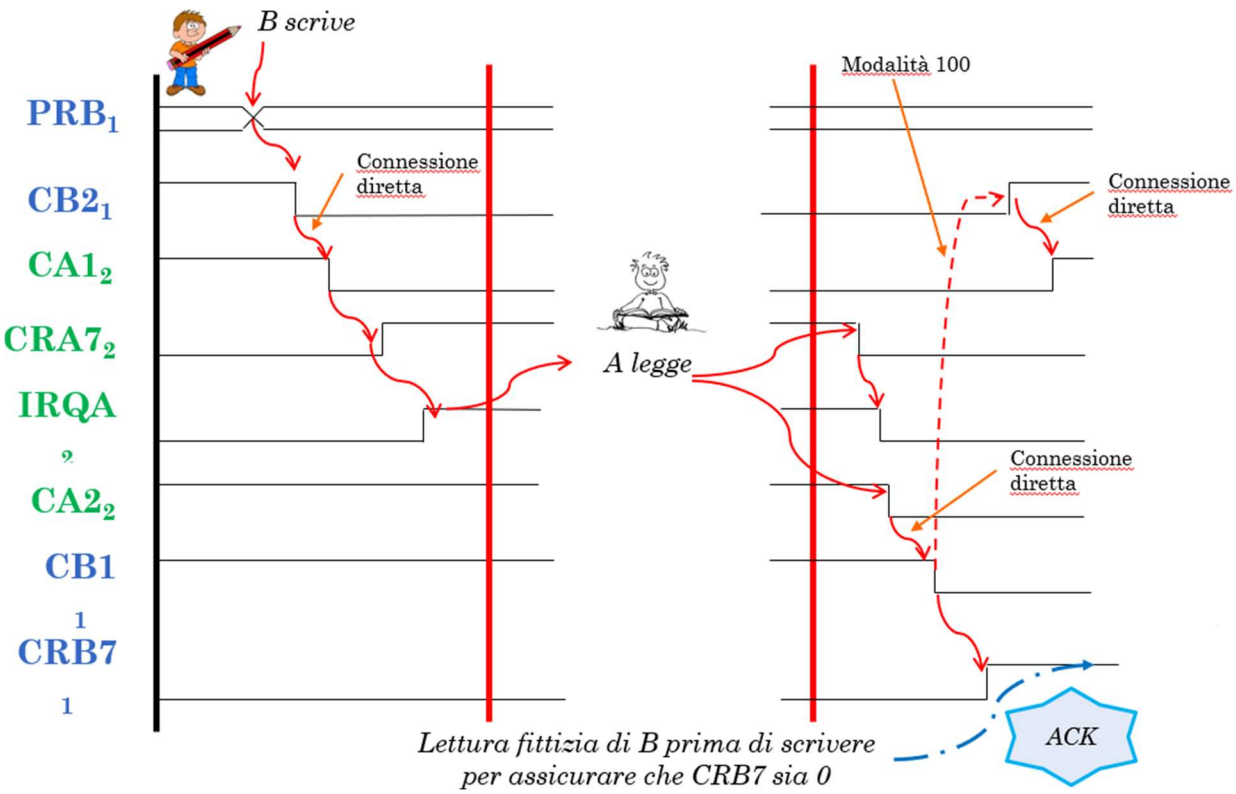


COLLEGAMENTI PIA

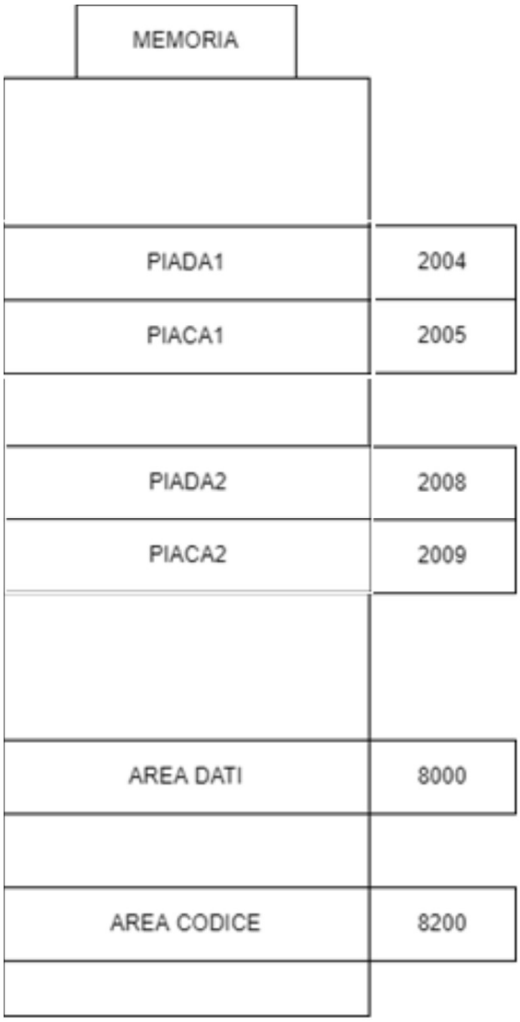


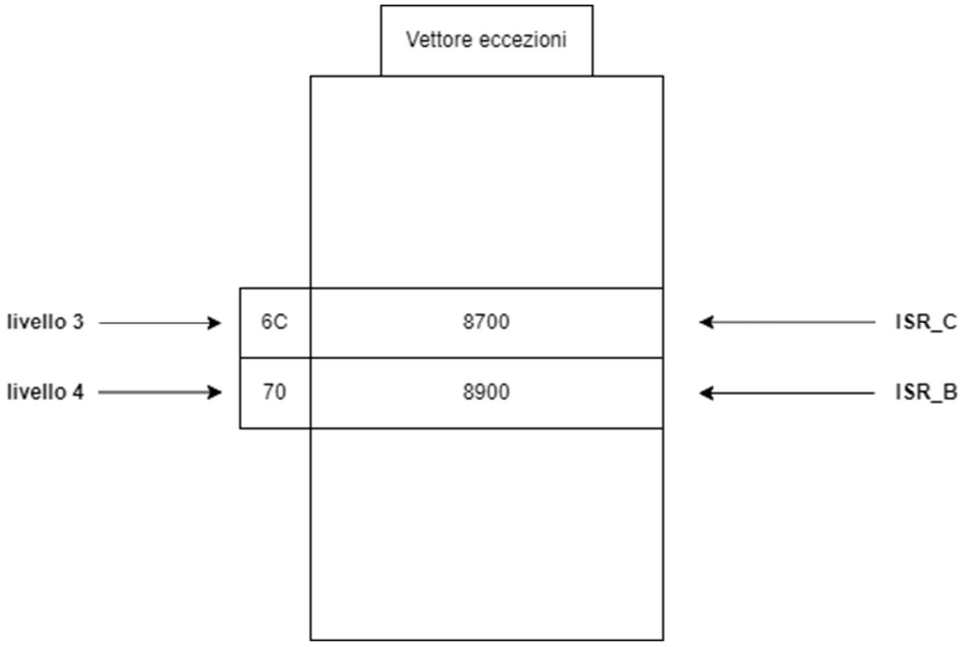

A
C
8 bit dato
CA1
CB1
Porto A
Porto B
CA2
CB2

PROTOCOLLO


B scrive
PRB1
CB21
CA12
CRA72
IRQA2
CA22
CB11
CRB71
Connessione diretta
Modalità 100
Connessione diretta
A legge
Connessione diretta
ACK
Lettura fittizia di B prima di scrivere per assicurare che CRB7 sia 0

MAPPA DELLA MEMORIA





PSEUDOCODICE

```
main:
    init_pia()
    mask_interrupt()
    loop

lastSend = 3
count_N = 0
fase = 1

ISR_C(){
    if(get_mutex()){
        func_c()
        if(wait_b==1){
            func_b()
            wait_b = 0
        }
        release_mutex()
    }else{
        wait_c = 1
    }
    rte
}

ISR_B(){
    if(get_mutex()){
        func_b()
        if(wait_c==1){
            func_c()
            wait_c = 0
        }
        release_mutex()
    }else{
        wait_b = 1
    }
    rte
}

func_b(){

    if(fase==1){
        if(N==0){
            N = get_byte()
            rts
        }
        b = get_byte()
        msg[count_N]=b
        count_N++
        if(count_N==N){
            fase = 2
            count_N = 0
        }
        rts
    }else if(fase==2){

        if(lastSend==1){
            wait_b = 1;
            rts
        }
        lastSend = 1;

        b = get_byte()
        msg[count_N]=b
        count_N++
        if(count_N==N){
            count_N = 0
            count_M++
        }
```

```
68.         if(count_M==T){
69.             fase = 3
70.             disable_interrupt()
71.         }
72.     }
73. }
74.
75.
76. func_c(){
77.     if(fase==1){
78.         wait_c = 1
79.         rts
80.     }else if(fase==2){
81.
82.         if(lastSend==2){
83.             wait_c = 1;
84.             rts
85.         }
86.         lastSend = 2;
87.
88.         b = get_byte()
89.         msg[count_N]=b
90.         count_N++
91.         if(count_N==N){
92.             count_N = 0
93.             count_M++
94.         }
95.         if(count_M==T){
96.             fase = 3
97.             disable_interrupt()
98.         }
99.     }
100.        }
```